Student Name

Student Email

Abstract

This document provides an example outline for assignments or papers for the course CS779. The format of this document is not required for the assignments in CS779 but is here to show an example of formalization and can be used as a template. The real purpose of this document is to show a good example of an organized document.

Assignment Template

MET CS 779 Assignment Template

MET CS 779 Assignment Template

Table of Contents

[1. Introduction 2](#_Toc8753632)

[2. Some Heading 1 2](#_Toc8753633)

[1) Some Sub-heading 1 2](#_Toc8753634)

[2) Some Sub-heading 2 2](#_Toc8753635)

[3. Some Heading 2 2](#_Toc8753636)

[1) Some Sub-heading 1 2](#_Toc8753637)

[2) Some Sub-heading 2 2](#_Toc8753638)

[4. Conclusion 2](#_Toc8753639)

[5. Revision History 3](#_Toc8753640)

[Appendices 4](#_Toc8753641)

[Appendix A 4](#_Toc8753642)

[Bibliography 5](#_Toc8753643)

# 1. Outline 3 main goals of allocation within my design

Goals 1: Allocating the data close to the locations where it used most.

Goals 2: Considering security implications in the data allocation decision.

Goals 3: Allocate an unreplicated fragment on the instance where it is most frequently used or where its presence will result in the best overall performance.

# 2. Design

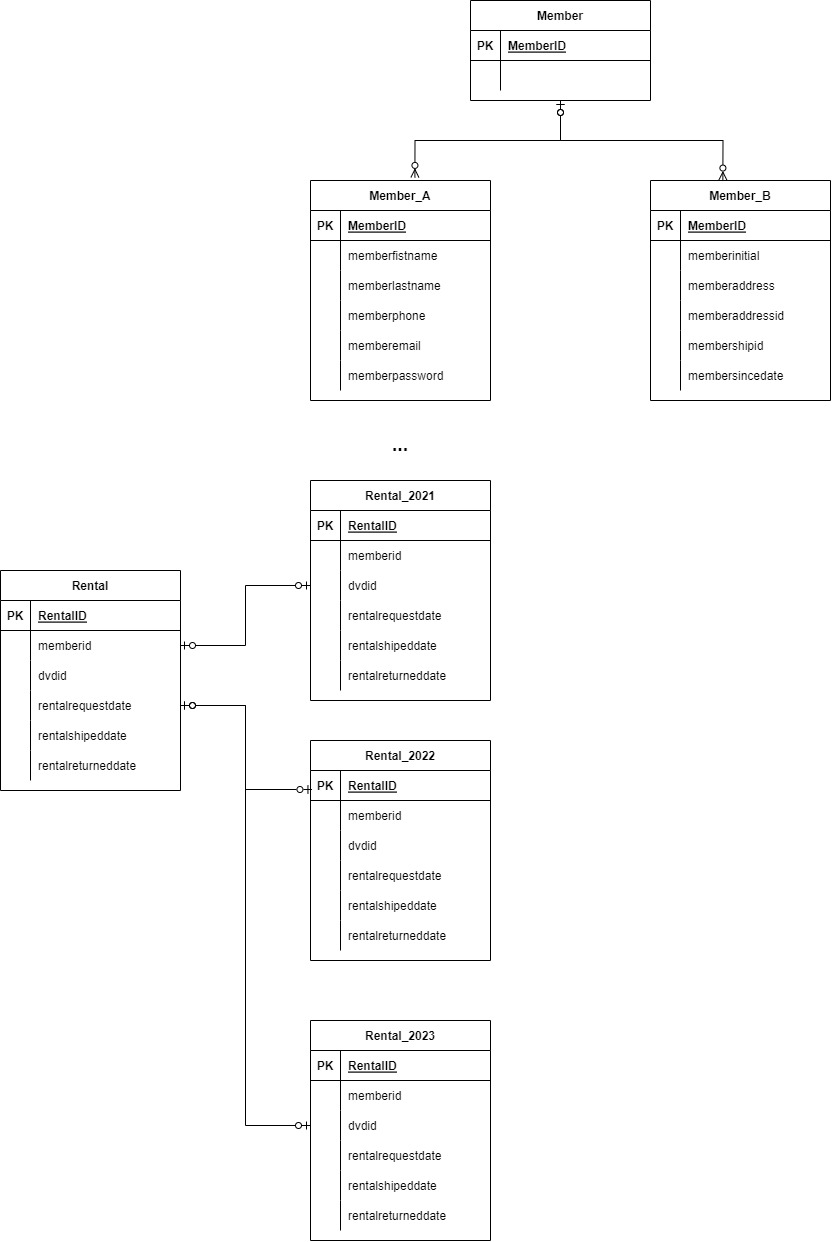
## Diagram (ERD) showing which tables and their columns if applicable are fragmented and how.

In Netflix database, I choose two tables: member and rental to make them fragmented.

For member table, I think it’s more applicable for vertical fragment. And their columns: memberfirstname, memberlastname, memberphone, memberemail and memberpassword are fragmented to member\_A table. And the remaining columns are segmented into the member\_B table.

Every fragmented table member\_A and member\_B both have the same primary key: memberID.

For rental table, I think it’s more applicable for horizontal fragment. Here I use rentrequestdate to split years and divide the records into different year’s subset rental table, such as rental\_2021, rental\_2022 and rental\_2023 and so on.



## Focus on elements which are not obvious

Because netflix has a lot of members, the basic information of members such as firstname, lastname, phone, email, password ,etc, are the information that is frequently used by the member login system;

The member's address information is only used when renting DVDs, and member since date and membership id need to be updated when purchasing membership. These information are not used very frequently;

Therefore, for the member table of Netflix, it is more suitable to use the vertical fragment solution.

For the Rental table, the rental data is generated by the members of netflix. The rental data accumulates more and more over time. The historical rental information has relatively few query and update requirements for netflix, and it can be said that there are almost no update operations. And recent and current rental information is what netflix pays attention to. Therefore, the rental table is processed by the horizontal fragment method.

## key advantage and disadvantage of the fragmented design

(1) key advantage:

Using vertical fragment to slice table member, it can avoid locating sensitive data such as member’s address and member’s membership information on instances where it could be accessed by individuals or programs that should not have access to it.

Using horizontal fragmentation of table rental, it can solve the problem of slow access of rental table which storing huge rental data over time. It fragment the whole table based on years and help reduce the storage size of the data. And history rental data will be stored in other fragmentations. Access to recent and current rental data will be greatly improved.

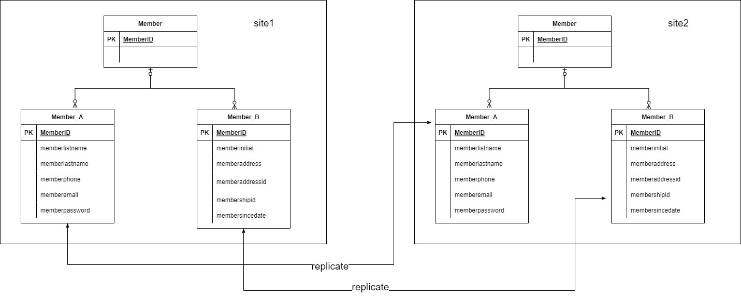
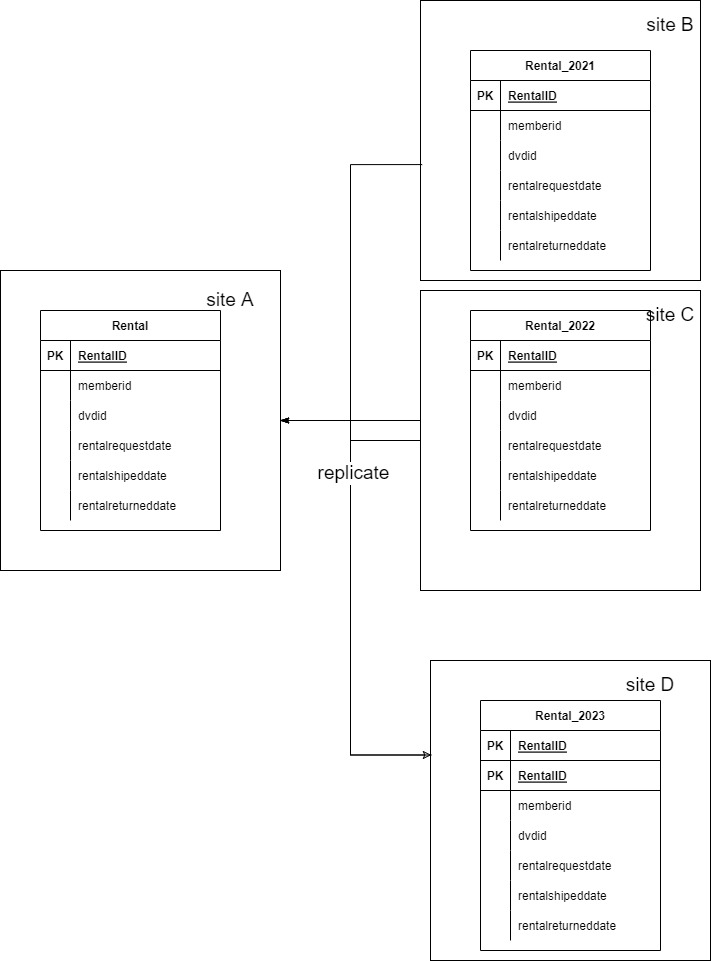
(2) key disadvantage:

Using vertical fragment to slice table member, when a member want to purchase membership, it must use memberId to join member\_A table and member\_B table. It will bring the cost of joining tables, especially when customer\_A and customer\_B are located on remote database instances, the cost of network traffic will increase.

Using horizontal fragmentation of table rental, when a member's rental information needs to be queried, his rental information may be distributed over many fragments, increasing the cost of distribution transparency.

# 3. DDBMS architecture

## Diagram a distributed replicated/fragmented design



## Explanation

In my replication design, the table rental and member are distributed. For member table member\_A and member\_B table may located in different database instances.

And for table rental\_2021, and rental\_2022, rental\_2023 may located in different database instances.

For member table, if member\_B information was added in site 2, then the data will be replicated to site 1.

For rental table, the rental is replicated table, and other subsets are unreplicated table. So when the subset table in site B,C and D has added new data, it will replicated into site A.

Here I choose partially replicated databases. For the rental table and member table are both incremental over time.

In partially replicated databases, some fragments may be replicated at sites where they are frequently needed. For example, infrequently updated tables such as those that represent member’s ensitive data may be replicated to all sites where they are joined. Sometimes tables that hold critical data such as the rentals may be replicated to protect against data loss. When replicating frequently changing fragments be sure to estimate and measure the network traffic and performance degradation to maintain the replicas.

## Explanation

(1)key advantage

The security risks of member data can be well guaranteed.

And the data loss can be reduced.

(2) key disadvantage

The cost of replicating data from one site to another one need lager cost.

# 4. Some research of real-world relational database

Amazon Aurora is a MySQL- and PostgreSQL-compatible relational database built for the cloud that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.

Amazon Aurora uses a distributed, fault-tolerant, and self-healing storage system that can scale up to 128TB per database instance. It enables high performance and high availability, supporting up to 15 low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three Availability Zones (AZs).

Amazon Aurora is up to five times as fast as a standard MySQL database and three times as fast as a standard PostgreSQL database. Amazon Aurora is fully managed by Amazon Relational Database Service (RDS), which automates time-consuming administrative tasks such as hardware provisioning, database setup, patching, and backups.

For Amazon Aurora has advantages of high performance and scalability, high availability and durability, high security, so it can be used to implement my design of Netflix database.

# 5. Revision History

A history of things you added and why, not required but nice to have.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Version** | **Description** |
| Your Name | 01/09/19 | 1.0 | Initial Document Creation |
| Your Name | 01/15/19 | 1.1 | Added more headers to provide better example |

# Appendices

## Appendix A

# Bibliography

Ballve, M. (2014, January 18). Apps Rule The Phone, But The Mobile Web Is Still Alive And Well. Retrieved from http://www.businessinsider.com/html5-and-the-web-is-still-relevant-2014-1

Darlington, K. W. (2011). Designing for Explanation in Health Care Applications of Expert Systems. *SAGE Open*, 1-9.

Medlock, S. e. (2011). LERM (Logical Elements Rule Method): A method for assessing and formalizing clinical rules for decision support. *International Journal of Medical Informatics*, 286-295.

Whitten, J. L. (2007). *Systems Analysis and Design Methods.* New York: McGraw-Hill/Irwin.